Deep Learning Part 1

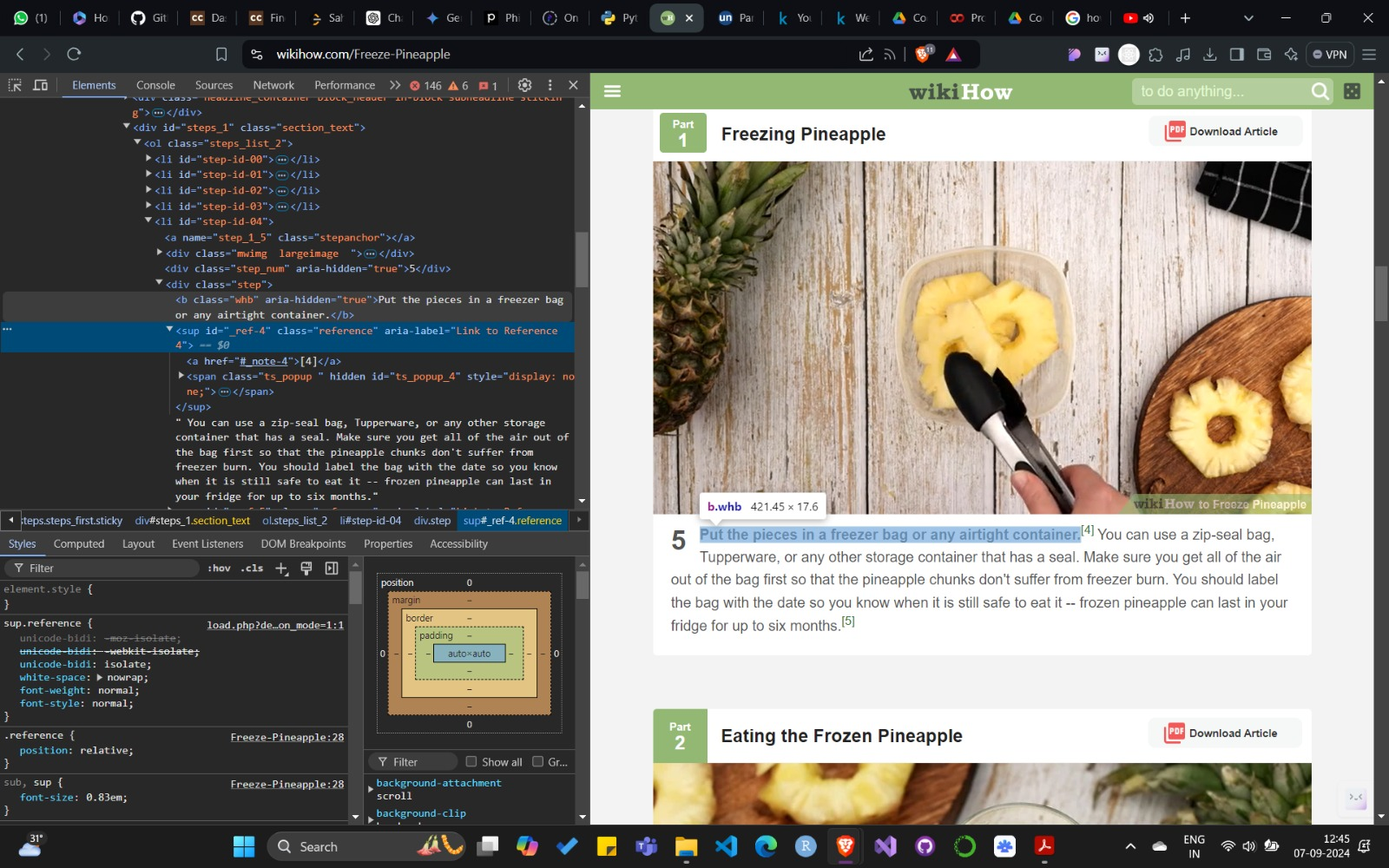
Section 1

**Web Scraping using the Library Beautiful Soup**

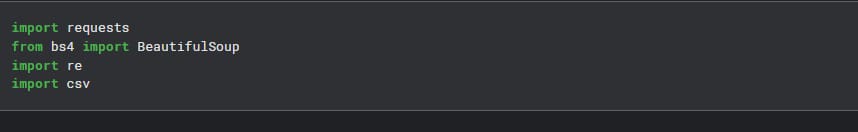
In the realm of data-driven applications, having a robust and relevant dataset is important. For a blog application, a well-curated and neatly labeled collection of articles that is well segregated can significantly enhance the content quality of the application. To achieve this, we are using **web scraping** using **BeautifulSoup** in Pythoninstead of collecting the data manually.

### **Step 1: Data Inspection**

We started by learning how articles in WikiHow are structured using dev tools. We had to understand what we actually require for our dataset which is the paragraphs and and subtitles from the article.



First we start by importing the necessary packages like requests (a HTTP library which is used to make HTTP requests of web services, from bs4 (Beautiful soup 4) importing the BeautifulSoup,then re for pattern matching and text manipulation.



Repeating the scrapping process for multiple process for multiple post inside a loop(eg, 4000)

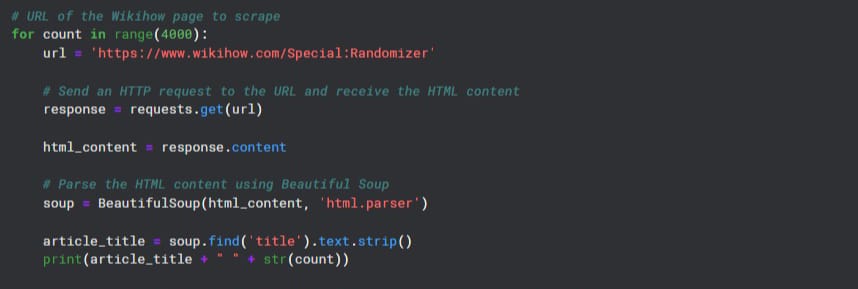
Using response to get content and BeautifulSoup to parse the html to separate the title and subheadings

**Step 2: Text Processing**

This script was designed to scrape WikiHow articles and store their content in a CSV file.

Here, we are selecting random articles from WikiHow by visiting the Special:Randomizer URL. An HTTP GET request retrieves the HTML content of each article.

BeautifulSoup is used to parse the fetched HTML. We extract and clean the article title and display it alongside a counter to keep track of the articles being processed.



In this block, we identify the subheadings and paragraphs from the content. We clean the text by removing unnecessary tags like <span> and stripping out formatting elements such as bold tags.

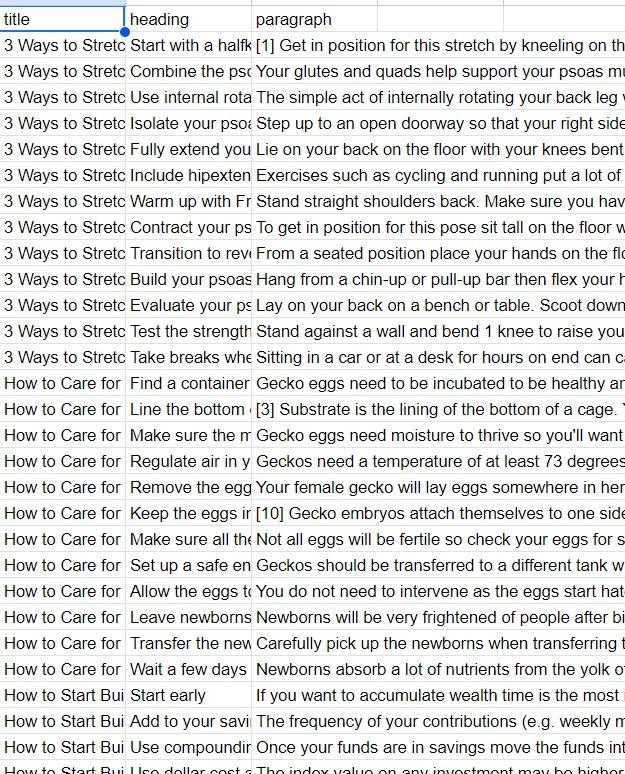


Once the subheadings and paragraphs are extracted, they are appended to a CSV file. Each row contains the article title, subheading, and corresponding paragraph, ensuring the content is well-structured.

* Clean the subheading (remove newlines, extra spaces, encode as ASCII).
* Remove unnecessary tags inside the div, leaving only the paragraph content.

### **Step 3: Building the Dataset**

The subheadings and paragraphs were saved into a CSV file, with each row containing the blog title, subheading, and corresponding paragraph. We repeated this process to scrape data for about 4,000 blog posts, ending up with a dataset of over 50,000 rows.(csv file with 50000 rows)



**Scalability and Limitations:**

Scraping a large number of articles comes with certain challenges and limitations. While this code scrapes 4000 articles, scraping at such a scale can lead to several issues:

* **Rate Limiting and Throttling:** Many websites, including WikiHow, impose rate limits or detect scraping activity after a certain threshold. They may block your requests if too many are made in a short period. To prevent this, it’s advisable to add time delays between requests.
* **Error Handling:** The website could return unexpected errors (e.g., 404 Not Found or 500 Internal Server Errors). Implementing error handling (such as retry mechanisms) ensures your scraper doesn't break when it encounters such errors.
* **Scalability:** For larger datasets, running such scripts for thousands of articles may consume significant bandwidth and time. Breaking the scraping process into batches or using distributed scraping frameworks like Scrapy could help manage large-scale scraping more efficiently.